International Research Journal of Management, IT & Social Sciences
Available online at https://sloap.org/journals/index.php/irjmis/
Vol. 6 No. 1, January 2019, pages: 113~121
ISSN: 2395-7492
https://doi.org/10.21744/irjmis.v6n1.769

The Sorted Nodes in Leveled DAG Division Algorithm and Critical Path on Processor

Mark Damon Gorn [a]
Matt Orchid Jack [b]
Conan Ballmon Enderson [c]

Article history:

Received: 09 July 2018
Accepted: 30 November 2018
Published: 31 January 2019

Keywords:

critical path;
division algorithm;
leveled DAG;
processor;
sorted nodes;

Abstract

Scheduling Algorithms, mostly List based static algorithms are considered for HDCS. Based on the algorithms, SNLDD, HEFT, CPOP, and implementation of SNLDD with Superior Performance Optimization Procedure are studied. In this paper, the outcome performance of the developed SNLDD algorithm is correlated with current existing algorithms mainly for HeDCSs. The comparative study between the proposed SNLDD algorithm with modified optimization procedure SPOP and HEFT with CPOP are evaluated based on the schedule length, speedup, efficiency of running programs and quality parameters with respect to memory in parallel computer systems has achieved a high speed up and fast execution time by SNLDD.

Author correspondence:
Mark Damon Gorn,
Harvard University, Cambridge, Massachusetts, United State.
Email address: gorn@harvard.edu

1 Introduction

Multiprocessor system in the Distributed Computing system is a system with more processors residing apart. Every Individual processor has its private, confidential memory, allowing security. Data Transmission is transferred through message passing between the processors. The processing elements interact with each and every processor in order to accomplish the common outcome.

Distributed Computing systems support the following aspects:

a) Providing efficiency in communication delay, balancing of load, distributed algorithms for huge messages.
b) Providing flexibility in terms of modularity, scalability, portability, and interoperability.
c) Computational Consistency that gives correct outputs irrespective of time at which the computation is finished for the identical input conditions.
d) Ensuring robustness in certain circumstances of security violations and also in flexibility failures.

[a] Harvard University, Cambridge, Massachusetts, United States
[b] Harvard University, Cambridge, Massachusetts, United States
[c] Harvard University, Cambridge, Massachusetts, United States

Figure 1. Distributed Computing Systems

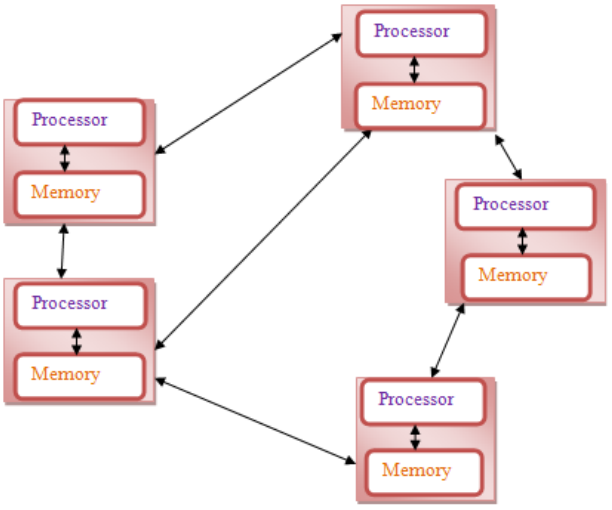

Figure 2. Sharing of results with different Processors

Task Scheduling

Task scheduling deals with the process of organizing tasks among processors and minimizes its length in scheduling, i.e, the time is taken to execute in every task corresponds to the starting time of the initial task. Queries generated on the outcome is the selection of tasks to processors. Real-time task scheduling originally deals with regulating of order in which the different tasks are to be reserved up for implementation by the operating system. Every operating system depends on more than one task, schedules to prepare the schedule of application of different tasks it process to execute. Each task scheduler is identified by the scheduling algorithm it operated. A massive number of designed algorithms for scheduling, real-time tasks have been refined. Real-time scheduling

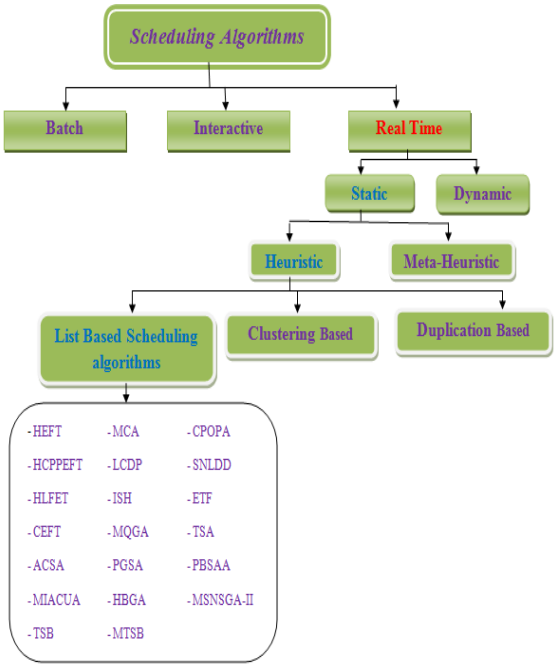

Figure 3. Scheduling algorithms classification

Literature Survey

Distributed computing systems are the number of an individual collectivity of processors interconnected through a tremendous system of connections that maintains the implementation of parallel operations (Bahnasawy *et al.*, (2011). The effectiveness of implementing various parallel operations on the systems desperately relies on the adopted approach to perform the tasks of the various parallel operations on existing processors (Yuan *et al.*, 2009). Every System in DCs, the execution of parallel computations in inter-processor communications is liable (Wang *et al.*, 2013). This burden exists whenever tasks are positioned to various processors in the transfer of data. Accordingly, in Heterogeneous distributed Systems generating high capacity task performances in parallel implementation are more critical (Arabnejad *et al.*, 2017). Such Conditions, an extensional arrangement among achieved speedup by automatic parallelization, the communicational inter mechanisms over systems, HeDCSs having scheduling algorithms by considering the different processors executing at different times of the same task. Some unreliable Scheduling arrangement in HeDCSs may restrict the effectiveness of the system when they perform in the very slow processors (Kessler, 1998). Mostly, Distributed Computing Systems in Scheduling algorithms have Static scheduling and Dynamic scheduling as two different existing task scheduling.

The primary Static scheduling algorithms, perform operations on the entire existing data for scheduling, for the framework of parallel operations, processing operations with respect to execution time, the cost among the tasks of communication is acknowledged prior (Kessler, 1998). To evaluate specific information various approaches are used (Shi *et al.*, 2006). In the parallel implementations, Static task scheduling concurrently has an activity of running processes at compile time (Venkateswaran & Mazumder, 1994).

Lately, a modernized algorithm known as the Longest Dynamic Critical Path was initiated (Hao *et al.*, 2010). Based on the algorithm, an improved attribute has identified the priorities accurately in HeDCSs of each task. The outcome of the LDCP is measured for comparison of two algorithms HEFT and the DLS (Kessler, 1998), algorithms related to performance.

This paper provides, a modern beneficial algorithm known as Sorted Nodes in Leveled DAG Division (SNLDD) to support HeDC systems with a defined total estimated number of processors for static implementation of task scheduling. The goal or achievement of using this algorithm is to obtain the large capacity scheduling of tasks with necessary to produce a tremendous performance in the Heterogeneous Distributing Computing Systems (Lin & Gen, 2018).

SLNDD is calculated with a parallel correlative study among SLNDD and the HEFT algorithm for execution. Based on these results, the SNLDD algorithmic approach excels the HEFT algorithm in conditions of schedule distance, time, performance, and characteristics of system behavior (Serna, 2008).

The SNLDD design approach has been changed by recommending a new process known as Superior Performance Optimization Procedure (SPOP) for the reduction of the sleek time by applying the idle time concurrently referring tasks to the processors by producing high-quality scheduling of tasks, and to reduce scheduling length. In addition, both updated algorithms are correlated and altered SNLDD algorithm confirms in excelling performance than transformed CPOP.

Problem Definition

Parallel operations are expressed by DAG for HeDCSs, in static task scheduling.
Direct Acyclic Graph is determined by considering tuple (T, E), T defines the collection of 'n' tasks and E defines the collection of 'e' edges. Every task, t_i corresponds to T in a parallel function, and every edge(t_i, t_j) corresponds to an order from the beginning of the constraint and a transfer of information among tasks t_i and t_j. With the condition edge E(t_i, t_j), the implementation of t_j T is implemented after t_i T completes its implementation. The task that is started first t_i as an edge (ti, tj) is represented as the parent to the sink task t_i, at the same time t_j is represented as a child of the t_i. Any individual task performing operations with no parents are named entry task, and any individual task performing operations with no children are named exit task. Most quantities of data that is transmitted from one task to another task (i.e., t_i to t_j) are related to every edge (t_i, t_j) [Hao, Q., Shen, W., Xue, Y., & Wang, S. (2010).].

Heterogeneous Distributed Computing Systems is determined by a group of m processors with P that has distinct result performances. Computation cost matrix (W) (i.e., n*m) keeps the implementation costs of all the 'n' tasks in m processors. Every component wi,j W produces the time of task processor with predicted performance. Now, every individual processor for the performance is simulated altogether linked. Information exchange among processors can take place through individual links; by this simultaneous execution of evaluation tasks and interactions among processors are approved (Zhu *et al.*, 2015). Computational costs of the tasks are acknowledged without changing. Especially, when task (t_i) computation costs at the respective processor (p_j) and processor (p_k). The communication cost between any two processors (p_k) and (p_i) rely on the network initialization at processors p_k and p_j further to the transfer time on the network. Processors (p_k) and (p_i) communication cost rely on processors (p_k) and (p_j) at network initialization and the time needed to communicate. Sender initially starts at some point in time and time at the receiver side is ignored when compared to the residing network time. The data transfer rate inside the network among the two processors predicted as constant and permanent. Thus, edges in transmission correspond to the data transferred from (t_i) to (t_j), separated by the network at a transfer rate and the network is united without failure of generality while achieving data transfer rate mostly in the inter-processor network. Accordingly, every single edge is equivalent to $d_{i,j}$ with respect to tasks on various processors at communication cost mostly scheduled. Considering, the connections identical processors among any two tasks are hired at zero. The Processor implements its individual tasks exclusively parents from the entire data applicable to exact processors; in such a case the task is ready to run and marked as running. The total run time of the parallel operations is allocated to achieve minimum runtime (Lam *et al.*, 2014).

2 Materials and Methods

The Sorted Nodes in Leveled DAG Division Algorithm (SNLDD)

One of the advanced task scheduling approaches known as SNLDD has been introduced and is related to partitioning graphs into different levels altogether concentrating on the need of priorities among tasks in the DAG. These tasks at every level are categorized into a list positioned based on computation size. These tasks are allocated to the first initiated processors arranged in precedence to the list. The computational size of every individual task is evaluated as follows:

$$S_j(n_i) = (w_j(n_i))_{p_f} + \{C_j[(n_i)_j, \sum_{k=1}^{t}(n_k)_{j-1}] + C_j[(n_i)_j, \sum_{x=1}^{q}(n_x)_{j+1}]\} \quad (1)$$

Where $S_j(n_i)$ describes the computation size pointing to a specific task (n_i) in the j^{th} level
i.e,$1 \leq j \leq R$, here R defines the number of levels in the graph and

$1 \leq i \leq T$, T defines the number of tasks in the schedule.

The primary part of the mentioned equation evaluates the performance task time n_i considering at processor p from j^{th} level with the high speed of the system.

The second part of the above-mentioned equation describes the sum of transmission among the task n_i in j^{th} level. Parent nodes in that $j\text{-}1^{th}$ level, and Childs nodes in $j+1^{th}$ level of communications.

Considering LDCP algorithm, tasks estimating the longest path based on the DAG. Every task performs lots of arithmetic evaluations because of repeated computations causing overheads in communications [6]. For this reason, the introduced SNLDD algorithm considers DAG algorithm as a base and use the graph by sub diving the levels of DAG and allocating each level task to processors. Thus, overheads are evaluated in the computations. In SNLDD algorithm, tasks in each level are partitioned by obtaining the sizes of the computation and sorted checking the priority, which is easy to classify. That simplifies SNLDD is more efficient when compared to the LDCP algorithm as a result, the time for selecting each task which was returned is allowed will be evaluated in every step.

Most often, after partitioning DAG graphs into subdivisions, tasks are allotted to each level, introduced SNLDD algorithm grows with higher efficiency compared to the LDCP algorithm. Following are some demonstrations for the algorithm:

a) It aims to restore the entire directions, operating time, tasks and communications at every appointing step that is not required to introduce SNLDD algorithm, overheads at execution time are removed in SNLDD algorithm.
b) Processors based on their computational size comforting not only adequate scheduling of tasks further accepts to develop the entire system of allocation of tasks basing on many characteristics like its cost for communications, reliance, and execution time at authorized tasks.
c) Partitioning every level tasks are based on computing size by managing scheduled tasks with substantial computation size and then tries to reduce the need among tasks.
d) Processors performing sleek time are reduced as long as DAG subdivision of levels and allocating to the processors.
e) The introduced algorithm is designed to have more effective when compared to other algorithms like LDCP, HEFT, and LCD.
f) Several different designs of extremely high currently used algorithms like sorting list, tree algorithms, and clustering algorithms are confirmed. Therefore, it is confirmed that SNLDD algorithm is a set of selected algorithms.

To overcome the repeated computations on each step of LDCP, SNLDD algorithm, the computation size of every task is executed only once in DAG, and restoring them by estimating on every step. This takes time in the evaluation of SNLDD algorithms for better calculations. $\Theta(m \times n\text{^}2)$ is the time complexity, where m describes a number of processors used in the scheduling algorithm, and n describes the number of tasks considered in the scheduling algorithm. The following figures 4 and 5 are the sample assumed DAG with computational costs.

Task	P1	P2	P3
1	12	14	7
2	11	17	16
3	9	11	17
4	11	6	15
5	10	11	8
6	11	14	7
7	5	13	9
8	3	9	12
9	16	10	18
10	19	5	14

Figure 4. Example of direct acyclic graph

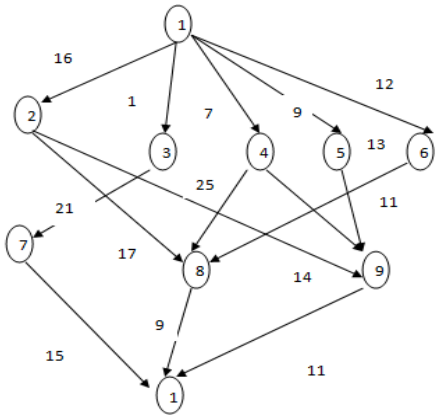

Figure 5. DAG with computation matrix

Implementing the Superior Performance Optimization Procedure (SPOP)

SPOP is known for a modernized optimization procedure with high-performance, extending itself to specify algorithms. It concentrates on the reliability of the tasks assigned to the processors by reducing its time considering scheduling policy:

a) The scheduling policy assigned to the tasks by the processors at insertion based assumes every feasible idle time on the processor is achieved data time slot that is similar and higher length to the computation/ implementation time. Precedence constraints among tasks are very required in performing.
b) The idle time is computed by exploring the ready state time by defining the length and inserted back to the final task that is scheduled on the processors.

HEFT (Heterogeneous Earliest Finish Time)

HEFT is one of the straightforward and best approaches in the static task scheduling both environments like heterogeneous and homogeneous even when few processors are considered. HEFT deals with crossing rather evaluating two phases: one is Prioritization phase and another is processor selection stage.
In the first stage of Prioritization: HEFT estimates the importance of the tasks by using the upward ranking ($rank_u$). An application operation is moving over in upward path and achieves entire list nodes with its rank with the service provided by the mean communication and performance cost. An obtained list is organized in systematized descending order of the $rank_u$. Every task with upward rank is defined as:

$$Rank(n_i)=W_i + max_{nj \epsilon succ(ni)}(C_{ij} + rank_u(n_j))$$

Wi describes the mean computation cost, succ(ni) is described as the immediate child of node ni, ci,j is described as the mean communication cost of node (i,j). If any two nodes have when two nodes obtain the similar rank value, it will prefer indiscriminately. The attained graph is moved from the end node to start node. The maximum rank value is similar to the end node:

$$rank_u\ (n_{exit})= W_{exit}$$

Critical Path on Processor (CPOP)

CPOP manipulates in two methods, namely upward ranking and downward ranking. CPOP calculates the rank by the value of every node by combining paired the ranking approaches rank + ranked and assign them to a queue. CPOP also has two levels of calculation: task prioritization level and task allocation level.
The first phase of task prioritization, every task is prioritized based on the rank value ($rank_u$ + $rank_d$) by use of transmission and computational costs that id obtained from DAG, Later, they are placed in a queue(in low order). CPOP performs Critical path to obtain the longest path from the start node to last node.
The second phase of task allocation, all the tasks are picked concentrating on maximum rank and chosen the minimum execution time of the task giving appropriate processor.

3 Results and Discussions

Result analysis is based on the attributes, namely: schedule length of the tasks, speedup of the processors and efficiency of the entire algorithm.
Comparison metrics of the algorithms SNLDD after modification through the performance of Superior performance optimization Procedure and with the Heterogeneous Earliest Finish Time with respect to Critical Path on Processor.
Schedule length finds overall execution time of a considered Direct Acyclic Graph (DAG), which is also known as makespan. Speedup defines the ratio of processed schedule length with the total number of processors. Efficiency gives the estimated by dividing speed with the total processors each time when they run.
Figures 7 and 6 are the schedule length with 10 tasks and the speedup with processors.
Figure 6 describes the speed of the processors in the distributed computing systems when 20, 40, 60, 80, 100 are considered. It shows that when the number of processors is where the speed is high. The figure thus concludes that the algorithm SNLDD with optimized procedure has given maximum result than the algorithm HEFT.
Figure 7 describes the schedule length of the considered DAG gives overall execution time. Here the comparison is done among the algorithms SNLDD without Optimization Procedure,i.e., only SNLDD, with SPOP, HEFT, and CPOP. Among all the algorithms the execution time of the modified SNLDD gives the fastest evaluation time.

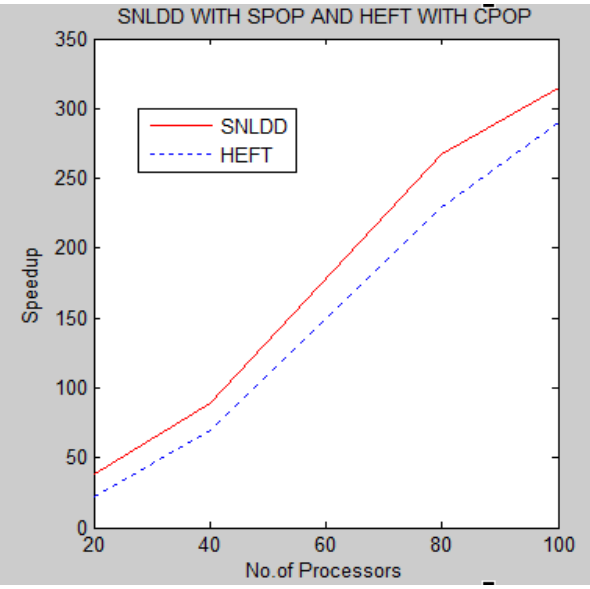

Figure 6. Speedup of the processors

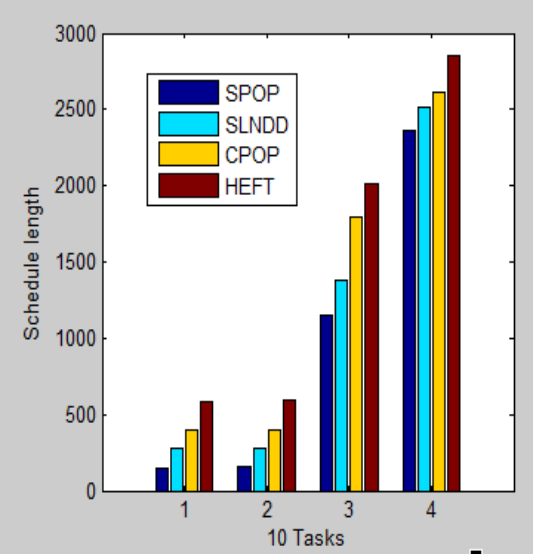

Figure 7. Schedule length with obtained by SPOP with SNLDD, SNLDD, CPOP, HEFT

4 Conclusion

Scheduling Algorithms, mostly List based static algorithms are considered for HDCS. Based on the algorithms, SNLDD, HEFT, CPOP, and implementation of SNLDD with Superior Performance Optimization Procedure are studied. In this paper, the outcome performance of the developed SNLDD algorithm is correlated with current existing algorithms mainly for HeDCSs. The comparative study between the proposed SNLDD algorithm with modified optimization procedure SPOP and HEFT with CPOP are evaluated based on the schedule length, speedup, efficiency of running programs and quality parameters with respect to memory in parallel computer systems has achieved a high speed up and fast execution time by SNLDD.

Conflict of interest statement
The authors declared that they have no competing interest.

Statement of authorship
The authors have a responsibility for the conception and design of the study. The authors have approved the final article.

Acknowledgments
The authors would like to thank the reviewer for their consideration the further process of the present paper. Thanks to the editor of IRJMIS for the valuable support, time as well as advice.

References

Arabnejad, V., Bubendorfer, K., & Ng, B. (2017). Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. *Future Generation Computer Systems*, *75*, 348-364. https://doi.org/10.1016/j.future.2017.01.002

Bahnasawy, N. A., Omara, F., Koutb, M. A., & Mosa, M. (2011). Optimization procedure for algorithms of task scheduling in high performance heterogeneous distributed computing systems. *Egyptian Informatics Journal*, *12*(3), 219-229. https://doi.org/10.1016/j.eij.2011.10.001

Hao, Q., Shen, W., Xue, Y., & Wang, S. (2010). Task network-based project dynamic scheduling and schedule coordination. *Advanced Engineering Informatics*, *24*(4), 417-427. https://doi.org/10.1016/j.aei.2010.07.001

Kessler, C. W. (1998). Scheduling expression DAGs for minimal register need. *Computer Languages*, *24*(1), 33-53. https://doi.org/10.1016/S0096-0551(98)00002-2

Lam, K. Y., Zhu, C. J., Chang, Y. H., Hsieh, J. W., Huang, P. C., Poon, C. K., & Wang, J. (2014). Garbage collection of multi-version indexed data on flash memory. *Journal of Systems Architecture*, *60*(8), 630-643. https://doi.org/10.1016/j.sysarc.2014.06.004

Lin, L., & Gen, M. (2018). Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications. *International Journal of Production Research*, *56*(1-2), 193-223. https://doi.org/10.1080/00207543.2018.1437288

Serna, M. (2008). Parallel Algorithms for Two Processors Precedence Constraint Scheduling: 2003; Jung, Serna, Spirakis. *Encyclopedia of Algorithms*, 627-629. https://doi.org/10.1007/978-0-387-30162-4

Shi, Z., Jeannot, E., & Dongarra, J. J. (2006, September). Robust task scheduling in non-deterministic heterogeneous computing systems. In *2006 IEEE International Conference on Cluster Computing* (pp. 1-10). IEEE. https://doi.org/10.1109/CLUSTR.2006.311868

Venkateswaran, R., & Mazumder, P. (1994). A survey of DA techniques for PLD and FPGA based systems. *Integration, the VLSI journal*, *17*(3), 191-240. https://doi.org/10.1016/0167-9260(94)90001-9

Wang, L., Khan, S. U., Chen, D., KołOdziej, J., Ranjan, R., Xu, C. Z., & Zomaya, A. (2013). Energy-aware parallel task scheduling in a cluster. *Future Generation Computer Systems*, *29*(7), 1661-1670. https://doi.org/10.1016/j.future.2013.02.010

Yuan, Y., Li, X., Wang, Q., & Zhu, X. (2009). Deadline division-based heuristic for cost optimization in workflow scheduling. *Information Sciences*, *179*(15), 2562-2575. https://doi.org/10.1016/j.ins.2009.01.035

Zhu, C. J., Lam, K. Y., Chang, Y. H., & Ng, J. K. Y. (2015). Linked Block-based Multiversion B-Tree index for PCM-based embedded databases. *Journal of Systems Architecture*, *61*(9), 383-397. https://doi.org/10.1016/j.sysarc.2015.08.002